A dispatching mechanism providing REMPLI Applications with QoS

Raul Brito, Ye-Qiong Song

A dispatching mechanism providing REMPLI Applications with QoS

Raul Brito
LORIA – INRIA - TRIO
Campus Scientifique – BP 239
54506 Vandoeuvre-lès-Nancy – France
raul.brito@loria.fr

Yeqiong Song
LORIA - UHP Nancy I - TRIO
Campus Scientifique – BP 239
54506 Vandoeuvre-lès-Nancy – France
song@loria.fr

Abstract

The main objective of the REMPLI project[1] is to develop a distributed infrastructure suitable for real-time monitoring and control of energy distribution and consumption. PLC technology has been chosen to form the main communication infrastructure.

The targeted applications possess different requirements in terms of Quality-of-Service (QoS) that should be taken into account by the underlying network. This comes to say that a traffic dispatching policy must be defined which guarantees minimum bandwidth utilization through periodic traffic and short end to end delay of aperiodic data request services.

1. Introduction

REMPLI PLC network is designed to cope with the specific characteristics of the communication channel. In [1] and [2] we have shown its good behaviours from the network performance point of view (minimized network transfer delay with small protocol overheads). However, when the REMPLI PLC is used for supporting REMPLI applications which share the same available bandwidth but require different quality of service (e.g. periodic transfer for network management services, short end to end delay for aperiodic data request services, urgent alarm handling, etc.), the REMPLI PLC must also give the possibility to serve those applications differently in order to satisfy their different QoS requirements.

The purpose of this paper is, on the one hand, to describe the traffic policy adopted in the REMPLI PLC under the name of Dispatcher, and on the other hand to prove its benefice in providing for the different application traffic with the differentiated QoS. In this sense, the Dispatcher ensures therefore the QoS mapping between an application and a REMPLI PLC network traffic class.

Figure 1 shows the general architecture of the system which is composed by three main components: Application Layer, Network Layer and Communication System. This paper will focus on the master Network Layer dispatcher mechanism as a means of guaranteeing a certain quality of service to the application data, as well as ensuring a stable network management system.

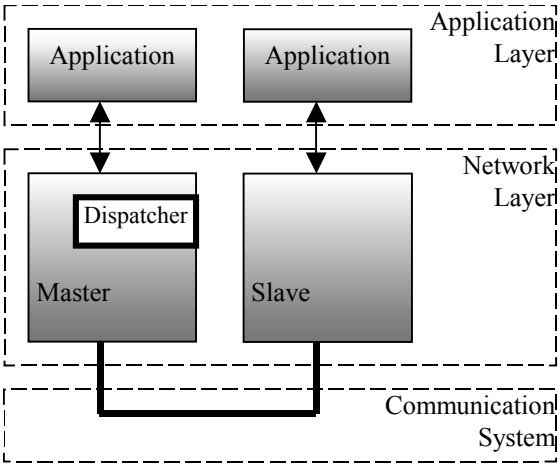

Figure 1 General architecture of the REMPLI system with emphasis on the dispatcher mechanism.

In section 2 we recall the application services defined within the REMPLI project and comment on the system constraints derived from such services. Necessary definitions of the system model are presented in section 3. The adopted network traffic scheduling policy is then explained and the principle of the dispatcher described in section 4. Section 5 presents a set of experimental simulation results that attest the interest of such a mechanism and finally a brief conclusion is given in section 6.

2. System Services and QoS Requirements

The available system services can be distinguished into application services and network management services.

[1] This work has been carried out as part of the REMPLI project (European program NNE5-2001-00825, www.rempli.org)

2.1. Application Services

Applications use the REMPLI network to support services to the end-users of the application. These applications services can be divided into 3 types: metering services, remote control and monitoring services, and file transfer service.

2.1.1. Metering Service

Provides meter reading requested by non-real time applications such as billing, consumption planning, prognostics and statistics. Although the applications do not have strict timing constraints, the metering data should be collected and the application response must be transferred within a reasonable time (few seconds).

2.1.2. Remote Control and Monitoring Services

Demands a real-time communication and offers the ability to monitor and control a process. The application, such as a SCADA server, remotely controls a specific device through command (e.g. command to meters for tariff change). On the other side, it can also supervise the operation status, including data request and the transmission of alarm messages indicating abnormal operation conditions.

For this service type, the network layer must be able to transmit critical packets that should be immediately sent to the slave node.

The fast delivery of events generated at the slave side is also covered by monitoring services. In this case, the system must guarantee a periodic poll of the network nodes in order to allow urgent events to be delivered at the application (at the master's side).

2.1.3. File Transfer Service

This service supports the transmission of a large amount of data (in the order of megabytes) through the network, working as a background task, relatively to other services.

2.2. Network Management Services

Network management services permit to maintain a correct control of the network, providing services such as logon and logout of nodes, information of network parameters, status and liveliness of nodes, etc.

These services can be considered both periodic (e.g. liveliness of nodes) and aperiodic (e.g. logout of slave).

One can also distinguish between the level of periodicity constraint affected to network management services.

3. System Model

The communication protocol model is master/slave based, in which the master takes the communication initiative by sending a packet, while the slave assumes a reactive position by responding with a corresponding confirmation. This packet can optionally carry data produced at the slave side by the application, such as a response to a previous request or alarm.

The network is composed by N nodes, divided into one master node and $N-1$ slave nodes. The access to the medium is assured by timeslot division multiplexing, where each timeslot allows the transmission of data in the size of a network layer packet. A packet sent by a node arrives at the neighbor nodes in a single timeslot. In order to reach a destination node, a packet sent by the source node can be reached directly (in a single timeslot) or is repeated by intermediary nodes (hereafter called repeaters).

One can distinguish packets into two types: aperiodic and periodic. Aperiodic packets contain data generated by the Application Layer. These packets are kept in queue buffers at the master (resp. slave) network layer before being delivered at the slave (resp. master) side of the network layer.

The network management at the master side periodically generates periodic packets for every slave, which permits to exchange information such as node status, liveliness, etc, and also guarantee a minimum bandwidth for every slave. The periodic packet i, noted by $P_i\left(C_i, T_i, \overline{D}_i, X_i\right)$, is characterized by the following attributes:

- C_i: execution time of packet i. C_i corresponds to the number of lapsed timeslots between transmitting a packet to the slave and receiving a confirmation packet at the master. The time between packet transmissions (which also includes dispatcher processing) is not taken into account, but can be easily added to the execution time by means of timing analysis tools.
- T_i: period of packet i. It means that in the following T_i timeslots an instance of packet i is transmitted.
- $\overline{D}_i$: defines the relative deadline of packet i. In our system we suppose $\overline{D}_i = T_i$, for all i.
- X_i: defines whether packet i has a hard or soft periodic constraint through a boolean variable, i.e, $X_i \in \{0,1\}$ with 0 meaning hard periodic and 1 meaning soft periodic. By hard periodic it means that in each period a packet transmission is necessary, while the missing of a deadline is not problematic in the case of soft periodic packet. This is equivalent to requiring a (m,k)-firm scheduling guarantee [3].

When explicitly referring to activation j of packet i, with $j \in \mathbb{N}$, the adopted index notation is i, j (e.g. the j^{th} activation of packet i execution time is noted $C_{i,j}$).

The execution time of packet i (C_i) results in the distance, in number of repeaters, needed by the downlink and uplink between master and slave, and also according to the strategy applied for adjusting the number of repeaters for downlink (defined as $r_{DL}(i)$) and uplink (defined as $r_{UL}(i)$) in case of a retry procedure. A packet retry occurs due when there is a transmission error or when the number of repeaters in the path has dynamically increased.

The worst case execution time is expressed by the following formula:

$$\overline{C_i} = r_{DL}(i) + r_{UL}(i) + \sum_{k=1}^{\max\_retries} f\left(r_{DL}(i), r_{IL}(i), k\right)$$

where the function f represents the applied strategy to increase the number of repeaters when in case of retries. This is a very pessimistic approach since in general no retries are needed most of the time. A coarser calculation is possible by eliminating the retries parcel:

$$\underline{C_i} = r_{DL}(i) + r_{UL}(i).$$

In the remaining of the paper, we suppose $C_i = \underline{C_i}$.

Whereas hard periodic packet attributes are static during each activation, soft periodic packets can have dynamic attributes. Briefly, this is caused by a transformation of packet periodicity from soft periodic to hard periodic. This subject will be discussed more extensively further on.

We note by H the current time of the system, in timeslots units since the start of the system.

4. The Dispatcher

The network dispatcher within the REMPLI system is a quality-of-service mechanism that plays an important role by permitting an optimal share of the network bandwidth among different traffic. Without it, applications could only expect a best-effort from the network (normally based in a first come first served criteria), which is unacceptable due to their timing constraints (e.g. freshness of data application at the slave).

The dispatcher (existing at the master side) is executed when the medium is currently free to initiate a new communication with a slave, for the next timeslot. Thus, the dispatcher must decide, at the master side, which is the next packet to be sent, among the different available packets, and based on the different system constraints.

The dispatcher knows exactly the existing aperiodic and periodic packets at the master side. Besides it, the slaves confirmation packets allow the dispatcher to derive some incomplete information about the packet status in the slave nodes. These three components are represented in Figure 2.

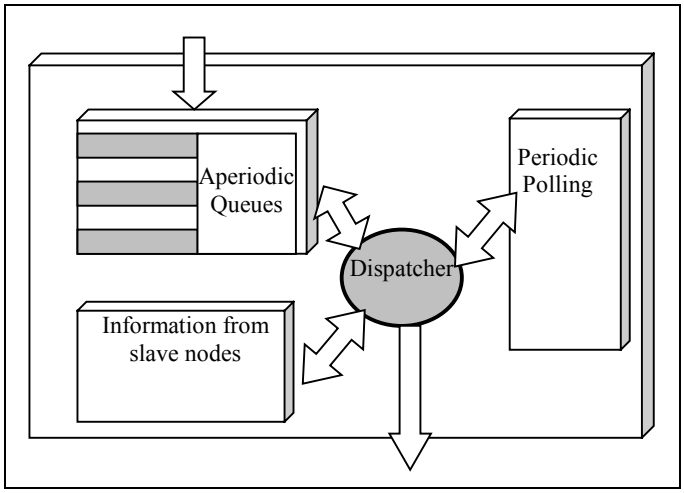

Figure 2 Network Layer component blocks that interact with the dispatcher.

The two following subsections (section 4.1and section 4.2) focus separately on aperiodic and periodic packet traffic and show how the dispatcher can determine which packet to choose. The integration of both traffics is covered in subsection 4.3.

4.1. Aperiodic Traffic

The aperiodic traffic existing in the REMPLI system can be generated in both master and slave side. Next, we describe the dispatcher criteria for deciding which aperiodic packet to send:

Packet Priority: the dispatcher uses the Network Layer prioritization scheme as a means for differentiating the importance of incoming application packets. These priorities can be divided into critical (noted priority 0), emergency (indicated as priority 1) and normal (priority 2). Priority 0 should be seldom used since it corresponds to a vital transmission of a command (e.g. shutdown command of a secondary substation due to fire alarm). Priority 1 is used by packets with emergency data. Although with a lower priority than the previous packets, but urgent enough not to be considered as regular data packet. Priority 2 is used by the normal traffic generated by the applications.

The choice of three priority levels permits an easier implementation efficiency of queue buffers, while still allowing a clear separation of the existing traffic at Application Layer which was described in section 2.1.

FIFO Principle: the First-In-First-Out (FIFO) behavior on the queue buffers should be respected. This denies the existence of starvation in queued buffers. Remark that more bandwidth efficiency approaches could be

envisaged (e.g. using data aggregation), but would result in an unfair bandwidth division between the slave nodes.

Slave Queues Information : based on the above requirement described above, the slave to master traffic flow is privileged, which means, from the Application Layer point of view, that it is more important to receive responses from slaves than to produce more new requests.

However, the dispatcher knowledge of slaves queue buffers is based on the information received in confirmation packets, which logically provides incomplete and not up-to-date information.

4.2. Periodic Traffic

The periodic traffic is generated at the master side and controlled by the dispatcher. This permits to guarantee a minimum bandwidth for network management and aperiodic packets from the slave nodes. Two approaches for guaranteeing the periodic traffic are given, both based in the Dual-Priority (DP) scheduling policy [4].

4.2.1. Dual-Priority Dispatcher

In DP, periodic packets possess two levels of priority: low and high level, whilst aperiodic packets are scheduled using a medium priority level. According to this, periodic packets can run immediately at a low level while there is no aperiodic traffic. In the presence of aperiodic traffic, a periodic task can only be sent when promoted to the high priority level, as late as possible. To calculate the time instant when a periodic packet is promoted, it suffices to calculate the response time for that packet. This will always guarantee the transmission of the periodic packet by its deadline. So, the promotion time L_i for periodic packet i scheduled according to DP is defined as:

$$L_i = \overline{D_i} - R_i,$$

where R_i is the response time and $\overline{D_i}$ is the relative deadline for the transmission of periodic packet i.

The calculation of the response time R_i of the periodic packet P_i is defined by $R_i = C_i + a_i^q$, according to the follow recursion [5]:

$$\begin{cases} a_i^q = 0 \\ a_i^{q+1} = \max_{j>i}(C_j) + \sum_{j<i}\left(\left\lfloor \frac{a_i^q}{T_j} \right\rfloor + 1\right) \cdot C_j \end{cases}$$

where the higher the packet index is, the lower its priority is. In practice it is only necessary to calculate a_i^q until convergence of the recursion value, instead of considering the infinite limit.

The set is feasible if the response time of every periodic packet is inferior to the relative deadline, otherwise it is unfeasible.

Based on this policy, our system must undergo a different approach for the calculation of the promotion time. It deals with the fact that the execution time of a periodic packet is dynamic, and therefore cannot be calculate off-line. Moreover, we deal with non-preemptive tasks, which are the case of the packet transmissions in the network and therefore have to wait until the transmission completion. We introduce the notion of promotion period α, which allows to guarantee that in at least one call of the dispatcher, every periodic packet can be successfully promoted without surpassing its deadline.

Thus, every time the dispatcher is called, it is necessary to verify on-line, for all periodic packets, if a promotion time is due. Packet P_i is promoted to the high priority level if the following inequality is true in the current period:

$$\left\lceil \frac{\mathrm{H}}{T_i} \right\rceil \cdot T_i < L_i .$$

Figure 3 shows an example of the method.

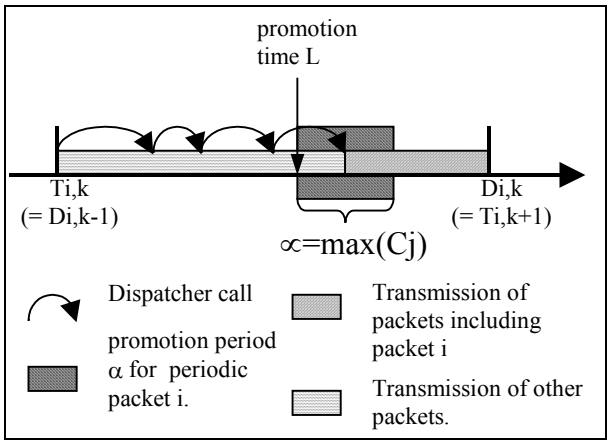

Figure 3 Adaptation of the Dual-Priority scheduling policy, with sucessive dispatcher calls until arrival into a packet promotion period.

As it can be seen, the dispatcher is called at least once during the promotion period of packet i, between L and $L + \alpha$, since α corresponds to the blocking factor due to non preemption (i.e, the maximum execution time of a packet, as considered in the definition of the response time).

4.2.2. Dual-Priority Dispatcher with Deadline Relaxation

The cost related to the computing of the promotion time (and consequent promotion period) for every periodic packet is very high in the last approach. This relates to the fact that the periodic packets have a deadline which ought be considered. Although this cost cannot be quantified at a design stage, we predict that the timing constraints in a implementation would be stringent, therefore in this section we propose an alternative approach.

We now show that by relaxing on the deadline constraint, it is possible to build a simpler and faster dispatcher, based on the Dual-Priority policy and still respecting the most important constraint in our system: the periodicity. The idea is to consider, for each periodic packet, its deadline as the promotion time L', that is, to have $L' = T$. In this way, the high cost in calculating the promotion instant that was inherent in the previous approach is inexistent since we have a static promotion time.

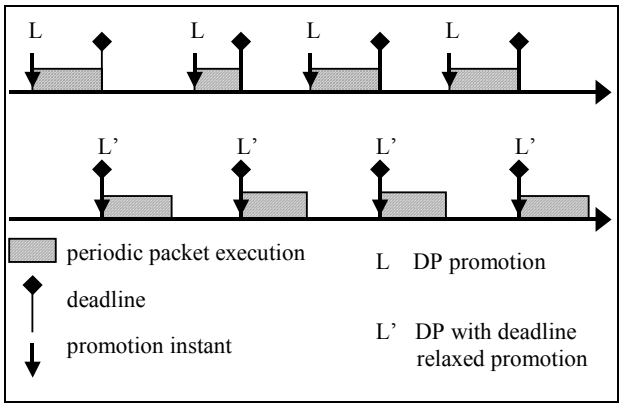

Figure 4 Representation of DP scheduling policy with imposing deadline in the upper time axis, and relaxation of the deadloine constrain in the lower time axis.the deadline constraint.

In Figure 4 an example of both approaches is depicted for execution of the same periodic packet. The upper time line represents the previous approach where the promotion time is calculated based on the deadline of the periodic packet. The lower time line represents the approach where the promotion time is dictated by the arrival of its deadline. In this case, since several periodic packets can have the same promotion time, in the worst case, a periodic packet P_i can have a maximum bounded jitter of $T_i + \alpha$. However, it still accomplishes the needed sense of periodicity.

This approaches also influences on the allowed maximum charge of the periodic traffic. That is, the relaxation of the deadline constraint permits the increase of the periodic packet to a maximum charge of 1. Thus, one only needs to guarantee that:

$$\sum_{i=1}^{M} \frac{C_i}{T_i} < 1,$$

where M is the number of periodic packets.

4.3. Components Integration

The purpose of this section is to detail on how the dispatcher will manage both periodic and aperiodic traffic of the system.

According to the requirements described in the first part of the article (section 2), the packet traffic precedence is defined in Figure 5, for every time that the dispatcher mechanism is called.

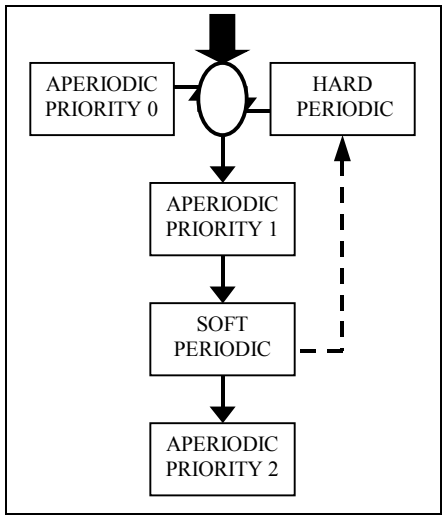

Figure 5 Representation of the precedence of the different types of traffic defined at the dispatcher. Every call of the dispatcher starts in the black arrow. The dashed arrow represents the promotion of periodic packets from soft to hard periodicity constraints.

A Round-Robin mechanism exists between aperiodic packets of priority 0 and hard periodic packets. This allows to maintain a correct management of the network (through the hard periodic packets) for one part, and to allow critical aperiodic packets to immediately be transmitted for another part; without creating a network monopolization by any of them.

Afterwards the dispatcher verifies the existence of aperiodic packets of priority 1, followed by soft periodic packets and finally aperiodic packets of priority 2. This order allows aperiodic packets of priority 1 to have a higher priority than soft periodic packets, since these last have lower periodic constraints. Nevertheless, the soft periodic packet can be temporarily upgraded into a hard periodic packet in order to guarantee the completion of the current activation, since it regains an higher priority than the aperiodic packets with priority 1. This process is

regulated by the X_i attribute of the periodic packet i, for the current activation of the packet.

5. Simulation Scenarios

We simulated the two approaches described in section 4. The first approach is based on the dynamic calculation of the promotion time of periodic packets, whereas the second approach is based on a static promotion of the same type of packets.

The goal of the simulation is to derive response times of the aperiodic packets, that is, the time between being available to be sent and the actual sending time. It permits to evaluate the minimization of the delay of the aperiodic packets, at the master side.

The power-line communication system is emulated by the Physical Layer Emulator [6] developed by iAd.

The comparison will be done in a ring network with one master and nine slaves, where the master sends aperiodic packets to all slaves in a uniform manner, but always respecting the following periodic traffic:

- $P_0\left(C_0, 255, 255, 1\right)$.
- $P_a\left(C_a, 3840, 3840, 0\right)$, with $a = 1..9$.
- $P_b\left(C_b, 378, 378, 1\right)$, with $b = 10..18$.

5.1. Single Aperiodic Queue

In this scenario, there is a single aperiodic queue of priority 2 (normal), with a buffer size of 40 packets. Packets are generated every 5 to 40 timeslots, for both the DP and DP relaxed deadline dispatcher approaches. The results are given in Figure 6 (delay of transmission of the aperiodic packets) and Figure 7 (percentage of acceptance of the packets into the queue buffer). Notice that the cause of non acceptance of a packet is consequence of a queue buffer overflow condition.

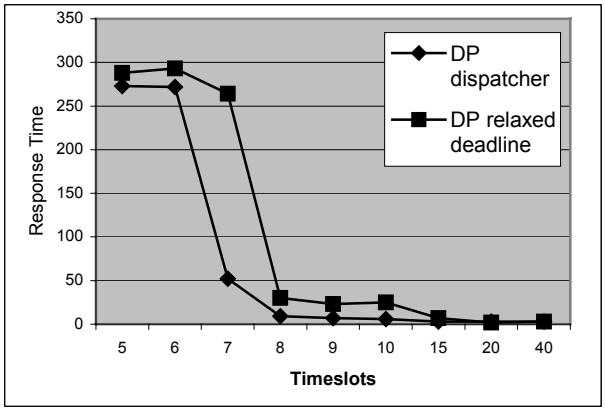

Figure 6 Response times of aperiodic packets generated at several timeslot units rates, noted in the x-axis, for both dispatcher approaches.

The percentage of packet acceptance in the queue buffer is high, while the difference between the two approaches is low.

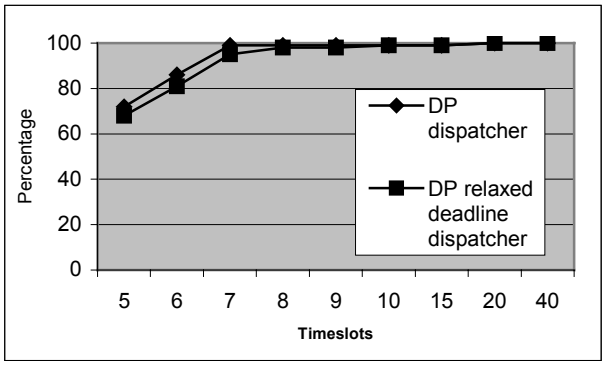

Figure 7 Percentage acceptance of aperiodic packets in the queues buffers for both dispatcher approaches.

Relatively to the packet acceptance in the queue buffers, it can be verified its similitude, although the DP dispatcher approach is slightly better.

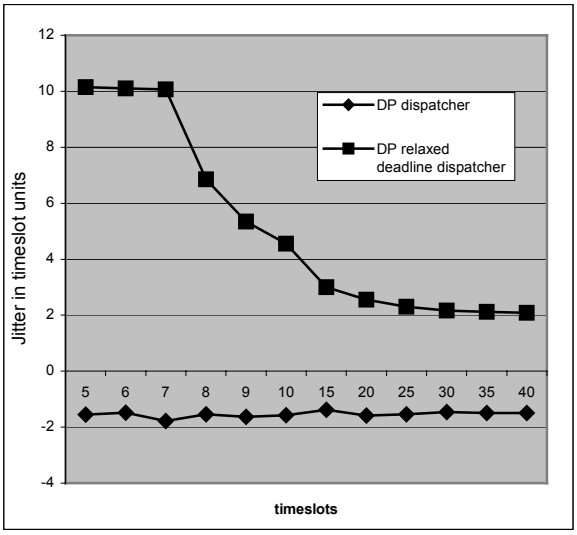

Figure 8 Jitter of the periodic packets for both dispatcher approaches relatively to the deadline, where jitter is equal to zero, with a single aperiodic queue.

Figure 8 shows that the DP dispatcher approach respects the periodic packet deadlines, while the DP relaxed deadline dispatcher approach does not respect, but the jitter is not very important, especially when the aperiodic traffic load decreases.

5.2. Double Aperiodic Queues

The simulation scenario with two aperiodic queues (Aperiodic Priority 1 and Aperiodic Priority 2 queue buffers) has the purpose of verification of the promotion strategy of periodic packets. In this way, aperiodic packets in the first queue (Aperiodic Priority 1) have a higher priority than soft periodic packets. However, in the case of missing deadlines, these soft periodic packets are promoted to the hard periodic table, with a higher priority than any aperiodic queue, which allows not to miss the deadline again.

Note that each queue buffer has a size of 20.

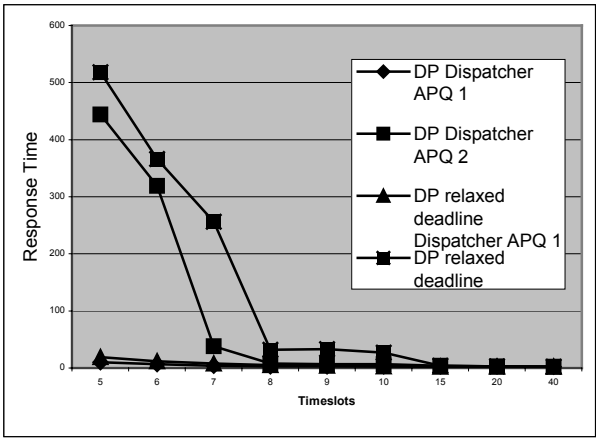

Figure 9 Response times of the aperiodic packets generated at several timeslot units rates, for both dispatcher approaches.

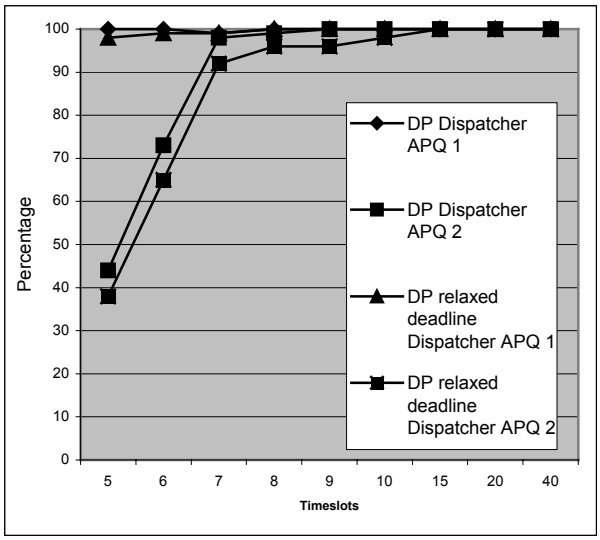

Figure 10 Percentage acceptance of aperiodic packets in the two queue buffers for both dispatcher approaches.

The introduction of the promotion from soft to hard periodic level shows that with a high load (aperiodic packets generated every 5 to 8 timeslots), the aperiodic priority 2 suffers greatly, in both approaches, while aperiodic priority 1 still maintains a reduced delay of transmission (check in Figure 9). Again DP dispatcher approach performs better than the DP relaxed deadline approach. The same high load provokes a lower acceptance of aperiodic packets (one out of two packets rejected) as it can be observed in Figure 10.

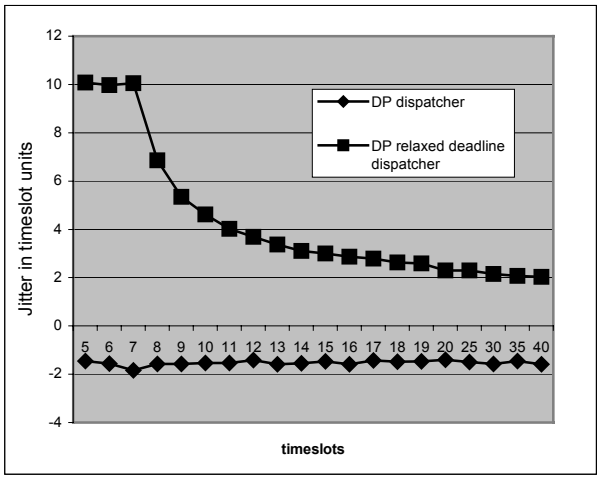

Figure 11 Jitter of the periodic packets for both dispatcher approaches relatively to the deadline, where jitter is equal to zero, with two aperiodic queues.

Relatively to Figure 11, which represents the jitter of periodic packets, with aperiodic packets generated for aperiodic queues 1 and 2, we verify the same behavior as

in the single aperiodic queue scenario. Thus, we can say that what influences the jitter is the load of the aperiodic queues in the system and not its priority for the deadline relaxed dispatcher approach.

These simulations show that the DP dispatcher approach produces better results, but the DP relaxed deadline dispatcher still allows to maintain a good performance relatively to the first approach, while decreasing the computation time for the calculation of the promotion time of the periodic packets.

Nevertheless, when comparing the simulation speed of computations of both approaches, the relaxed deadline dispatcher provides a considerable faster computation.

6. Conclusion

This paper proposes the dispatcher as a network layer mechanism that provides a quality of service to the applications in the REMPLI system. Based on the Dual-Priority scheduling policy, two dispatcher approaches are proposed in order to guarantee periodic traffic constraints while minimizing the end to end transmission time of aperiodic packets.

The comparison between both approaches shows that the relaxation of the deadline constraint permits to obtain reasonable approximation to the deadline strict approach while reducing the computation time, which is a crucial factor for deploying the system. We expect to produce more detailed information about the performances of both approaches as the dispatcher mechanism is being implemented in the scope of the REMPLI project.

References

[1] G. Bumiller, L. Lu, Y.Q. Song, “Analytic performance comparison of routing protocols in master-slave PLC networks”, ISPLC 2005, Vancouver, Canada, April 6-8 2005.

[2] R. Brito, G. Bumiller, Y.Q. Song, “Modelling and simulation of a SFN based PLC network”, ISPLC 2005, Vancouver, Canada, April 6-8 2005.

[3] J. Li, Y.Q. Song, F. Simonot-Lion, "Schedulability analysis for systems under (m,k)-firm constraints", WFCS2004, Vienna (Austria), Sept.22-24,20

[4] R. Davis, A. Wellings, “Dual-Priority Scheduling”, RTSS95, Pisa, Italy, 12 May 1995.

[5] A. Koubaa, Y.Q. Song, “Evaluation and improvement of response time bounds for real-time applications under non-preemptive Fixed Priority Scheduling”, International Journal of Production Research, 15 July 2004.

[6] G. Bumiller, “Power-Line Physical Layer Emulator for Protocol Development”, ISPLC 2004, Zaragoza, Spain, March 2004.